Optimal Multicast Group Communication

Zhibin Zhou and Dijiang Huang
Arizona State University

***Abstract*— Many IP multicast based applications, such as Pay-TV, Multiplayer games, require controlling the group memberships of senders and receivers. One common solution is to encrypt the data with a session key shared with all authorized senders/receivers. To efficiently update the session key in the event of member removal, many rooted-tree based group key distribution schemes have been proposed. However, most of the existing rooted-tree based schemes are not optimal. In other words, given the $O(\log N)$ storage overhead, the communication overhead is not minimized. On the other hand, although Flat Table scheme [1] achieves optimality [2], it is rather dismissed due to the vulnerability to collusion attacks.**

In this paper, we propose a key distribution scheme – EGK that attains the same optimality as Flat Table without collusion vulnerability. EGK also support dynamic subgroup communication initialized by each group members (imagine a virtual chat room in the multicast group). Additionally, EGK provides constant message size and requires $O(\log N)$ storage overhead at the group controller, which makes EGK suitable for applications containing a large number of multicasting group members. Moreover, adding members in EGK requires just one multicasting message. EGK is the first work with such features and out-performs all existing schemes.

I. Introduction

IP multicast is used to distribute data to a group of receivers efficiently. A Datagram addressed to the multicast group, identified by a Class D IP address, will be delivered to all group members. Efficiency can be achieved because datagrams need to be transmitted once and they traverse any link between two nodes only once, saving the cost of sender as well as network bandwidth.

To scale to a large receivers population, IP multicast group is open and multicast enabled receivers can freely join or leave by sending an IGMP message to neighbor routers. The IP multicast model has been described as "You put packets in at one end, and the network conspires to deliver them to anyone who asks". Thus, senders can not restrict receivers access to multicast data. In the mean time, the receivers have no way to authenticate the sender's identity and ensure the authenticity of multicasted data.

Many IP multicast applications require secrecy [3] and authenticity [4], [5] of transmitted data. Pay-per-view TV requires controlled subscriber membership and web feed of stock information requires authentic data source. Moreover, some applications, like multiplayer on-line games, dynamic conference, etc., where members send and receive simultaneously, requires both data privacy and source authenticity.

Furthermore, membership of the multicast group may change dynamically as members may join or leave the groups at any time. Despite the dynamics of legitimate member set, data senders want to preserve backward secrecy and forward secrecy, which are defined as follows:

- *Backward Secrecy*: new joined group members must have no access to past multicast communication;
- *Forward Secrecy*: revoked group members[1] must have no access to future multicast communication;

Multicast Group Key Distribution (MGKD) [6], [7] addresses the security problem in open network environments. MGKD restricts the group membership by encrypting the data with a symmetric group key (GK) shared among group members. As the members may join or leave the group dynamically, it is very critical to ensure only legitimated group members have the update-to-date GKs at each point in time, which is achieved by GK rekeying. Quite commonly, at the event of membership changes, a centralized group server generates a new fresh GK and distributes the rekey message to all legitimate members. All members in the group need to be attentive to the rekey messages to update their GK.

Particularly, removing members poses the scalability problem for rekey operation. To illustrate this problem, we can consider a set of clients L are removed from a group G. Rekey message should be generated and distributed to each of $G \setminus L$ remaining clients. Thus, the problem of removing any arbitrary set of members L can be transformed to the problem of communicating with a subgroup of any arbitrary $G \setminus L)$ members in a scalable and secure way.

A. Storage-communication-optimality

Before introducing the Storage-communication-optimality, we denote the *Encrypted Stream* as encrypted messages decipherable by one or a combination of secrets. For example, in nested encryption [8], the ciphertext $\{M\}_{K_1,K_2,K_3}$ is can be decrypted only if the decryptor has K_1, K_2 and K_3 in the same time.

To facilitate the rekey operation, auxiliary secrets need to be pre-distributed to group members. One naive way is to set a unique, long term key for each group member as advocated in [9] and [10]. To revoke a set of group members L, $|G \setminus L|$ messages are required to distribute a new GK. Thus, the communication cost is $O(N)$. In this example, each member can decrypt 1 encrypted stream.

We can trade storage space for communication cost in another extreme case. As there exists 2^N subsets of group members in the multicast group G of size N, group members in the same subset are distributed a unique subset key. Hence, every group member stores 2^{N-1} subset keys. When revoking group members L, the group controller only needs to multicast one message, i.e. a new random GK encrypted by the subset key shared by the subset $G \setminus L$, since each group member in

[1]Here we mean that a revoked group member is not eligible to participate in the group communication for a given multicasting group.

$G \setminus L$ already stores subset key of $G \setminus L$. In this example, the communication overhead is $O(1)$; while the storage overhead id $O(2^N)$. As one can see from this extreme case, each group member can decrypt 2^{N-1} encrypted streams. Obviously, $O(2^N)$ storage overhead makes this solution infeasible and some trade-offs between storage and communication overhead are needed.

To balance the communication and storage overhead, rooted-tree based key distribution schemes have been proposed, such as [7], [11], [3], [12], [13]. In these schemes (illustrated in Figure 1), each member is distributed $\log N$ secrets. In [2], the authors proved that assigning $\log N$ secrets to each member is the information theoretical optimal storage strategy when group size is N.

Despite the optimality in storage, most existing schemes including [7], [11], [13] are not optimal in communication and only improve efficiency marginally. This is because, given the $\log N$ distributed keys, each group member can only decrypt $\log N$ encrypted streams as explained later in Section III-C. Intuitively, as shown in the previous two extreme cases, the more encrypted streams each member can decrypt, the less communication, in terms of number of messages, overhead required for the rekey operation.

The storage-communication-optimality condition describes the balanced optimality condition between the storage and communication overhead incurred by group rekeying. In a storage-communication-optimal MGKD scheme, each group member can use their $\log N$ pre-distributed keys to decrypt maximized number of encrypted streams, i.e. $2^{\log N} - 1 = N - 1$ encrypted streams.

Flat Table (FT) scheme [1] (illustrated in Figure 2) achieved minimal the communication overhead [2] with $O(\log N)$ storage overhead and is storage-communication-optimal.Despite its optimal efficiency, FT scheme is vulnerable to collusion attacks, in which multiple removed members combine their pre-distributed secrets to decrypt updated GK to compromise forward secrecy. In this paper, we are presenting a scheme that is storage-communication-optimal and immune to collusion attacks.

B. Dynamic Subgroup Communication

In existing tree based MGKD schemes, there is only one communication group (the group protected by the session encryption key) and group members can not initialize an arbitrary subgroup communication in a ad hoc manner. This is because the group key is distributed in a one-to-many manner, i.e. only group controller has all auxiliary keys to distribute session key to any group members. Although each group member has some shared secrets, the auxiliary keys in Figure 1, there are still some unreachable members in the tree. For example, in Figure 1, M_1 is not able to initialize a secure channel with M_7 and M_8.

Let's consider the following example: an on-line multiplayer game. From the system's perspective, all subscribed players are in the communication group and a centralized server is in charge of the memberships. People can join or leave the game dynamically and real time game date should only be accessible by current subscribed players. From players' perspective, on the other hand, it is sometimes desirable to form subgroups in an ad hoc manner. For example, a team of players to finish common tasks want to set up a virtual chatting room. Unfortunately, existing Multicast Group Key Distribution schemes can not support this application nicely. Broadcast Encryption schemes, on the other hand, allow each user to encrypt a message to a set of receivers. However, Broadcast Encryption schemes are often very intensive, in terms of communication, storage and computation overhead. Also, there lacks mechanism to revoke a member's capability to broadcast encrypt messages [2].

C. Our Contribution

In this paper, we propose an Efficient Group keying (EGK) scheme to achieve non-colluding, storage-communication-optimal group key management. Moreover, EGK supports dynamic subgroup communication and each member can setup a secure conference with other members in an ad hoc way.

In EGK, a group controller (GC) is responsible for key generation and distribution and the group data are encrypted by a GK. Each group member (GM) is assigned a unique n-bit ID. For each GM, GC also generates and distributes a set of $n = \log N$ secrets, which are one-to-one mapped to the bits in the GM's ID. Note that, although different GMs may share common bits in their IDs, the pre-distributed secrets are different generated by using different random numbers. As a result, different GMs cannot combine their secrets that are masked by different random numbers. We denote the set of pre-distributed secrets as a GM's private key.

Whenever GMs are removed from the group, GC will multicast an encrypted key-update message. Only the remaining GMs are able to recover the message and update GK as well as their private keys. To achieve *storage-communication-optimality*, we use the similar method of Flat Table scheme. A minimized boolean function in the form of sum-of-product-expression (SOPE) is calculated based on the IDs of remaining GMs, in order to minimize the number of encrypted key-update messages. A remaining GM can combine n pre-distributed secret shares in his/her private key to decrypt a key-update message.

EGK is the first work that achieves storage-communication-optimality with constant message size and immune to collusion attack. It outperforms existing group key management schemes in terms of communication and storage efficiency. We must note that in [14], the authors utilized the ciphertext policy attributed based encryption (CP-ABE [15]) scheme to implement FT so that it is secure against collusion attack. Although their solution is similar to EGK, we adopt a different approach to improve the communication efficiency in comparison with using CP-ABE directly. As mentioned in [14], the size of each message is very large and grows linearly on the number of

[2]In the original introduction of Broadcast Encryption, only one centralized trusted server can perform encryption. Thus, the revocation is simple. Recent researches greatly elevate Broadcast Encryption by allowing all users to perform encryption such that any subset of users can decrypt the message. But, there is no centralized authority that controls user memberships.

attributes in the access policy [14], [15]. In EGK, the message is substantially reduced to a constant size.

Based on the storage-communication-optimality, EGK also supports dynamic subgroup communication efficiently. In existing MGKD, group members can only participate in the overall group communication protected GK, which is distributed in one-to-many manner. EGK allows each GM to initialize a subgroup communication with any subset of GMs in many-to-many manner. The number of required messages is minimized. Only GMs within this subgroup can securely communicate with each other.

Overall, the main contributions of EGK are presented as follows:

- With any number of removing GMs, the number of encrypted key-update messages is information theoretically minimized to $O(\log N)$.
- The size of each message in encrypted key-update message is constant.
- The communication overhead of adding GMs is $O(1)$, i.e., only one multicast message is required.
- The storage overhead of GC and GM is $O(\log N)$ even if GC does not store IDs of GMs.
- EGK is collusion resistant and provides forward and backward group key secrecy.
- EGK supports dynamic subgroup communication efficiently.

D. Paper Organization

The rest of this paper is organized as follows. We describe related works in Section II. Section III presents system models used in this paper. We present detailed EGK construction in Section IV. In Section V, we discuss the performance of EGK scheme and the comparison of several existing group key management schemes. We also analyzed the security of EGK in Section VI. Then, we discussed the scalability issue of EGK and further reduction of computation overhead in Section VII. Finally, we conclude our work in Section VIII.

II. Related Works

Multicast key distribution schemes have been investigated intensively in past two decades. Some of the works include but not limited to [16], [12], [17], [18], [19], [20], [7], [21], [22], [23], [24], [25]. Due to the richness of related research, we cannot list all the related work in this area. We refer to [26], [27] as two excellent surveys.

The rooted-tree structure (see Figure 1 and Figure 2) is constructed such that each group member is assigned as a unique leaf node in the tree. Every node in the tree, including leaf and non-leaf nodes, is assigned a unique auxiliary key. Each group member is predistributed a set of auxiliary keys that are along the path from the leaf to the root, in which the root associated auxiliary key can be used for the entire group. Using rooted-tree based solutions, an auxiliary key can be shared among a partition of members, and a member can be involved in multiple partitions. Typically, the a-ary rooted-tree based solutions require $O(\log_a N)$ storage overhead for each member [12], where N is the group size. The rooted-tree

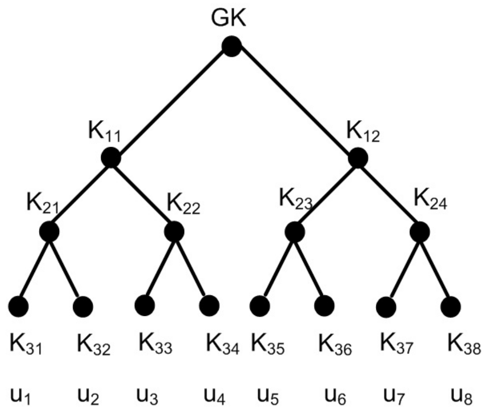
GK
K_{11}
K_{12}
K_{21}
K_{22}
K_{23}
K_{24}
K_{31} K_{32} K_{33} K_{34} K_{35} K_{36} K_{37} K_{38}
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8

Fig. 1. Illustration of two kinds of tree structures. Tree (a) is used in non-flat table schemes and tree (b) is used in flat table scheme.

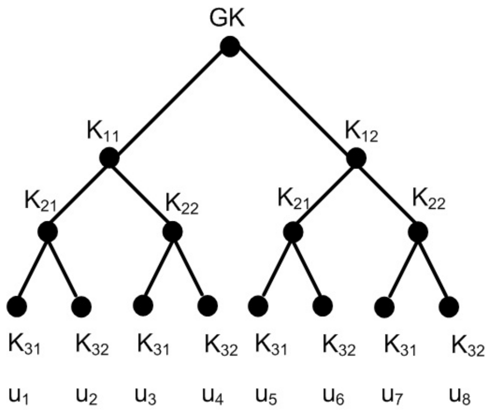
GK
K_{11}
K_{12}
K_{21}
K_{22}
K_{21}
K_{22}
K_{31} K_{32} K_{31} K_{32} K_{31} K_{32} K_{31} K_{32}
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8

Fig. 2. Illustration of two kinds of tree structures. Tree (a) is used in non-flat table schemes and tree (b) is used in flat table scheme.

based multicast group key distribution scheme can be divided into two categories: Non-Flat-Table schemes (Figure 1) and Flat-Table schemes (Figure 2).

Non-Flat-Table include most famous rooted-tree based schemes, such at OFT [11], LKH [7], and ELK [13]). One important feature of these schemes is there are a^i distinct secrets at level i in the key distribution tree as illustrated in Figure 1. In other words, each subtree in the level i is distributed a unique secrets. We note that the secrets there are not necessarily just predistributed keys [7]. They may be generated using one-way hash function [11] or pseudo random number generator [13].

Non-Flat-Table schemes only improves the efficiency marginally. This is because, in these solutions, based on the $\log_a N$ pre-distributed auxiliary keys, each group member can merely participate in $\log N$ partitions, as illustrated in Figure 1. We will explain this point later in Section III-C.

Flat-Table schemes [1], [28] adopt a slightly different construction of balanced key trees, as illustrated in Figure 2. In Flat-Table schemes, each group member is issued a unique binary ID $b_0b_1 \dots b_{n-2}b_{n-1}$ of length n. In addition to the GK, group server generates $2n$ auxiliary key encryption keys (KEK) $\{K_{i,b} | i \in \mathbb{Z}_n, b \in \{0,1\}\}$. A group member with ID $b_0b_1 \dots b_{n-2}b_{n-1}$ holds KEKs $\{K_{i,b_i} | i \in \mathbb{Z}_n\}$. The KEKs are organized in the key distribution tree in Figure 2, where each level corresponding to one bit position in a user's ID. Thus, at each level in the Flat-Table key distribution tree, there are exact 2 distinct pre-distributed keys, which map to a bit positions in a group member's ID. For example, in the Figure 2, member with ID 011 is predistributed $\{K_{11}, K_{22}, K_{32}\}$. In Flat-Table, the number of partitions each group member can participate is maximized to $2^{\log N} - 1 = N - 1$.

Despite its efficiency, Flat-Table schemes are vulnerable to collusion attacks since FT solutions simply adopt the symmetric key solutions. For example, GMs 001 and 010 can decrypt ciphertexts destined to other GMs, e.g., 011, 000, by combining their symmetric keys that are mapped to their bit positions. To prevent the collusion attacks, Cheung et al. [14] proposed CP-ABE-FT to implement the FT using CP-ABE. CP-ABE-FT utilizes a periodic refreshment mechanism to ensure forward secrecy. The periodic refreshment methond has several drawbacks: 1) if the ID of a revoked GM is re-assigned to another GM before the refreshment, the revoked GM can regain the access to group data and then the group forward secrecy is compromised; 2) outsiders can impersonate GC to disturb the rekey process by sending CP-ABE [15] ciphertext. More importantly, message size of CP-ABE-FT is linearly growing [14] and, thus, the communication overhead is actually $\log^2 N$. The strategy of EGK reducing communication overhead for GM leaving is similar to that of FT scheme, which is optimal. As a contrast, EGK features collusion resistance and a constant message size and communication overhead is $\log N$.

Broadcast Encryption (BE) was introduced by Fiat and Naor et al. in [38] and then followed by [29], [30], [31], [32], [33], [34], [35], [36]. In BE a broadcaster encrypts a message for some set of users who are listening to a broadcasting channel and use their private keys to decrypt the message. Compared with traditional one-to-one encryption schemes, BE features superior efficiency. Instead of sending messages encrypted with each individual recipient's public key, the broadcast encrypter broadcast one encrypted message to be decrypted by multiple recipients with their own private keys.

Although existing BE schemes [29] always features small or constant ciphertext, the number of public key or private key each user needs to perform encryption or decryption are linear on the max number of non-colluding users in the system. In the case of the BE scheme is fully collusion-resistant, the number of public/private key each user needs to store equals to the number of users in the system.

In the existing BE system with N users, each user $i \in \{1, \ldots, N\}$ is generated a public key PK_i and a private key SK_i. To encrypt a message to a set of users $S \subseteq \{1, \ldots, N\}$, the encrypting algorithm takes input of the set of public keys for all recipients $\{PK_i | \forall i \in S\}$ and output the ciphertext. To decrypt a message, the decrypting algorithm takes input of the private key SK_i of user i and the set of all public keys $\{PK_i | \forall i \in S\}$ to recover original message.

III. System Models And Background

A. Notations

the notations used in this paper is listed below:

Symbols	Descriptions
G	the Broadcasting Group Includes All GMs
L	a Subset of GMs
u	a GM
B	Bit-Assignment
S	Set of Bit-Assignments
GC	Group Controller
GM	Group Member

B. Communication Model

The communication model of EGK is based on IP multicast. All Group Members (GM) belong to a multicast group $G = \{u_1, u_2, \ldots, u_{|G|}\}$. Each GM u can send or receive diagrams encrypted by GK. The multicast group is associated with a trusted server,referred as Group Controller (GC), responsible for managing the membership. When one or more GMs are removed from the group, GC multicast the key update message and only remaining GMs can decrypt the message and update their keys. Each GM can initialize a secure subgroup communication with any subset of GMs. The subgroup traffic is also multicasted to whole group while only a designated subset of GMs can decrypt the data.

C. Storage-communication Optimality Condition

To better illustrate storage-communication optimality, we present a simple example of existing solutions under the considerations *storage-communication-optimality condition*. The rooted-tree structure (see Figure 1 and Figure 2) is constructed such that each group member is assigned as a leaf node in the tree.

Non-Flat-Table key distribution tree is shown in Figure 1. Three auxiliary non-root keys are assigned to group member u_2: K_{11}, K_{21}, and K_{32}. Note that combining multiple keys does not create a new encrypted stream as members holds K_{21} is true subset of members holds K_{11} and members holds K_{32} is true subset of members holds K_{21}. Using these auxiliary keys, u_2 can decrypt 3 encrypted streams:

Encrypted Stream (Key)	Accessible Members
K_{11}	$\{u_1, u_2, u_3, u_4\}$
K_{21}	$\{u_1, u_2\}$
K_{32}	$\{u_2\}$

As for Flat-Table case in Figure 2, 3 non-root keys are distributed to the group member u_2: K_{11}, K_{21}, and K_{32}. Using these auxiliary keys, u_2 can decrypt in 7 encrypted streams:

Encrypted Stream (Key)	Accessible Members
K_{11}	$\{u_1, u_2, u_3, u_4\}$
K_{21}	$\{u_1, u_2, u_5, u_6\}$
K_{32}	$\{u_2, u_4, u_6, u_8\}$
K_{11} and K_{21}	$\{u_1, u_2\}$
K_{11} and K_{32}	$\{u_2, u_4\}$
K_{21} and K_{32}	$\{u_2, u_6\}$
K_{11} and K_{21} and K_{32}	$\{u_2\}$

Note that u_2 needs to combine (such as using XOR or nested encryption) the presented keys for each subgroup to secure the enumerated subgroup communications. In FT schemes, based on the $O(\log N)$ pre-distributed secrets, each group member can participate in $2^{\log N}-1 = N-1$ subgroups by using their predistributed auxiliary keys.

The authors in [2] showed that the assignment of $O(\log N)$ keys per group member is the best strategy for group communication schemes.

It can be proved that the communication overhead can be reduced by increasing the number of encrypted stream that each group member belongs to. We can also observe that, given the $\log N$ predistributed secrets, a group key management scheme attains optimality only if the number of subgroups each group member can participate is maximized to $2^{\log N} - 1 = N - 1$. One example of optimal group key rekeying scheme is Flat Table (FT).

With maximized number of supporting encrypted streams, the GC or a GM can communicate with an arbitrary subgroup of GMs with minimized messages. For example, using Figure 1, to create a subgroup containing $\{u_2, u_4\}$, a group controller can encrypt the subgroup key SK_{24} using key encrypting keys (KEKs, a.k.a, auxiliary keys) K_{32} and K_{34} in parallel and multicasted two encrypted messages. However, using Figure 2, the group controller can encrypt the subgroup key SK_{24} using a combined KEK $K_{11} \oplus K_{32}$, which are only known to group members u_2 and u_4. In [2], the authors showed that using FT schemes presented in Figure 2 provides the optimal solution in terms of reducing communication overhead for group key management. However, FT suffers collusion problems, which prevent it from being used for secure group key management.

An MGKD scheme that is optimal on both storage and communication if and only if the following condition is satisfied:

Definition 1: Storage-communication-optimality condition: given the $\log N$ pre-distributed secrets for a group size of N, each group member can combine any of the $\log N$ secrets to decrypt $2^{\log N} - 1 = N - 1$ different encrypted streams.

D. Attack Models

Firstly, we assume that the symmetric encryption algorithm and one-way hash function used in this paper is secure. Also, we assume that the Discrete Logarithm Problem (DLP) on both group $\mathbb{G}_0$ and $\mathbb{G}_1$ is intractable. Finally, the GC is well guarded and trustable. In addition, our security analysis will focus on collusion resistance, forward secrecy, and backward secrecy.

The attackers' goal is to reveal broadcasted data without authorizations from GC. In particular, we can consider the attacking scenarios in the following cases:

1) *Collusion Attacks*: Multiple GMs combine their pre-distributed secrets to decrypt the ciphertext not intended to them. No particular example of this attack is that when multiple GMs are revoked from the group, they try to collude to recover the secrets of some valid GMs to continue reveal group data. Another example is that when a secure conference is held among a subgroup of GMs, some GMs excluded from the conference try to recover the secrets of an GM in the conference so that they can listen to the conference.
2) *Breaking Backward Secrecy*: GMs try to reveal any group data that were transmitted before they joined the group.
3) *Breaking Forward Secrecy*: GMs try to continue reveal the group data that are transmitted after they left the group.
4) *Breaking the Group Secrecy*: Non Group Members try to reveal the group data transmitted over the open networks.

In all of these scenarios, we assume that attackers can receive and stores all transmitted messages. But, there is no such a compromised insider GM that works as a decryption proxy for attackers.

E. Bilinear Pairing

Pairing is a bilinear map function $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$, where $\mathbb{G}_0$ and $\mathbb{G}_1$ are two multiplicative cyclic groups with large prime order p. The discrete Logarithm Problem on both $\mathbb{G}_0$ and $\mathbb{G}_1$ are hard. Pairing has the following properties:

- *Bilinearity*:
$$e(aP, bQ) = e(P,Q)^{ab}, \quad \forall P, Q \in \mathbb{G}_0, \forall a, b \in \mathbb{Z}_p^*.$$
- *Nondegeneracy*:
$e(g, g) \neq 1$ where g is the generator of $\mathbb{G}_0$.
- *Computability*:
There exist an efficient algorithm to compute the pairing.

IV. Constructions of EGK

A. ID and Bit-Assignment

In EGK, each GM is associated with a unique binary ID: $b_0 b_1 \ldots b_{n-2} b_{n-1}$, where $n = \log N$. The ID is issued by the GC when a GM joins the group. Once the GM left the group, his/her ID can be re-assigned to other joining GMs.

We can use a logic literal, which we call *bit-assignment*, B_i or $\overline{B_i}$ to indicate the binary value at position i in a particular ID. B_i indicates the i'th bit of an ID is 1; $\overline{B_i}$ indicates the i'th bit of an ID is 0. For a group with N GMs, the length of an ID is $n = \log N$ and the total number of bit-assignments is $2n$; that is, two binary values are mapped to one bit position(one for value 0 and one for value 1). We call the set of all possible bit-assignments to be *Universe* U, which contains $2n$ bit-assignments.

A GM u is uniquely identified by the set of bit-assignments S_u associated with u's ID. Also, multiple GMs may have a common subset of bit-assignments. For example, in Figure 3, a GM u_1's ID is 000 and a GM u_2's ID is 001, $S_{u_1} = \{\overline{B}_0, \overline{B}_1, \overline{B}_2\}$ and $S_{u_2} = \{\overline{B}_0, \overline{B}_1, B_2\}$ and $S_{u_1} \bigcap S_{u_2} = \{\overline{B}_0, \overline{B}_1\}$.

In EGK, the GMs can be organized as leafs in a binary tree with each non-root node marked with a bit-assignment (Figure 3). Note that there are only $2n$ different non-root nodes in the tree and each level contains 2 nodes. This is fundamentally different from existing tree-based schemes in [11], [7], [13], where there are 2^d distinct nodes at level d. The ID of a GM can be represented by links from the root down to the leaf. Thus, any two GMs will have at least one bit-assignment different.

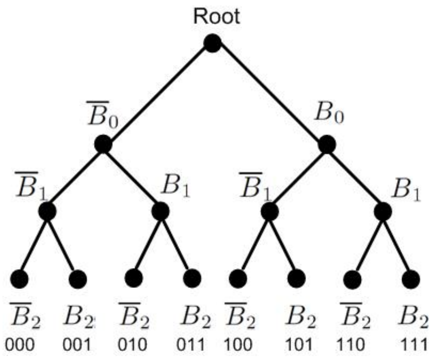

Fig. 3. An illustration of bit-assignments for a 3-bit ID space.

B. Group Setup

We describe how the GC sets up the multicast group. First, GC chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Also, GC chooses a publicly known one-way function H. Then, it chooses two non-trivial random numbers $\alpha, \beta \in \mathbb{Z}_p^*$. For simplicity, we can map the universe of bit-assignments U to the first $|U|$ members of $\mathbb{Z}_p^*$, i.e., the integers $1, 2, \ldots, |U|$. For each bit-assignment $B \in U$, GC chooses a non-trivial random number $y_B \in \mathbb{Z}_p$. We denote this set of $2n$ random numbers as

$$Y_B = \{y_{B_0}, y_{\overline{B}_0}, \ldots, y_{B_{n-1}}, y_{\overline{B}_{n-1}}\}$$

For each $y_B \in Y_B$, GC also generates the tuple $< e(g,g)^{\alpha y_B}, g^{\beta y_B} >$. We denote the set of $2n$ tuples as:

$$E_B = \{< e(g,g)^{\alpha y_B}, g^{\beta y_B} > | \forall y_B \in Y_B\}$$

GC publishes the group public parameter:

$$GP = \{\mathbb{G}_0, e, g, H, E_B\}$$

On the other hand, GC protect the group master key:

$$MK = \{\beta, g^\alpha, g^\beta, e(g,g)^\alpha, Y_B\}$$

C. GM Joining and Key Generation

When a new GM u joins the group, u needs to set up a secure channel with the GC using either a pre-shared key or public key certificates. GC then checks whether the GM is authorized to join in the group. Once the checking is passed, GC assigns a unique ID $b_{n-1}^u b_{n-2}^u \ldots b_0^u$ and a set of bit assignments S_u to u.

Once u is admitted to the group, GC runs key generation algorithm $\mathbf{KeyGen}(MK, S_u)$ (Algorithm 1) to generate private key SK_u for u, where MK is the group master key and S_u is the set of bit-assignments in u' ID. The algorithm first chooses a non-trivial random number $r \in \mathbb{Z}_p^*$. Then, it computes $g^{\frac{\alpha+r}{\beta}}$. Finally, for each bit-assignment $B \in S_u$, the **KeyGen** algorithm calculates a blinded secret share $g^{r y_B}$. The outputted private key

$$SK_u : \{D = g^{\frac{\alpha+r}{\beta}}, \forall B \in S_u : D_B = g^{r y_B}\}$$

If u is the first GM in the group, GC will generate an initial GK and sends the private key $\{SK_u, GK\}$ to the new GM u through a secure channel. If u is not the first joining GM, to preserve backward secrecy, GC generates another random key GK' and multicast $\{GK'\}_{GK}$. Each GM other than u can decrypt the message and replace GK with GK'. Finally, GC sends $\{SK_u, GK'\}$ to the new GM u through a secure unicast channel. In the *join* process, besides the unicast communication, GC only needs to multicast one message, i.e., $\{GK'\}_{GK}$. Thus, the communication overhead for GMs join is $O(1)$.

One important observation is that GC does not need to store the ID or private keys of any GMs. Thus, the storage overhead of GC can be significantly reduced to $O(\log N)$, since GC is only required to store the system parameters and master key.

Algorithm 1 $\mathbf{KeyGen}(MK, S_u)$

```
Randomly select $r \in \mathbb{Z}_p$;
Compute $g^{\frac{\alpha+r}{\beta}}$;
for each $B \in S_u$ do
    Compute $g^{r y_B}$;
end for
return
$SK_u : \{D = g^{\frac{\alpha+r}{\beta}}, \forall B \in S_u : D_B = g^{r y_B}\}$;
```

D. Encryption and Decryption

As we have mentioned, EGK allows GC and GMs to securely communicate with any subset of GMs. Whenever, GMs are removed from the group, GC needs to multicast a key update message to all remaining GMs, who will update their GK as well as private keys. On the other hand, GMs can initialize a secure subgroup communications with any subset of GMs.

In this section, we present how a GC or GM can encrypt a message with a set of bit-assignment S, so that only GMs whose IDs satisfy S can decrypt the message. For example, in a three-bit-ID group, if a ciphertext is encrypted by using bit-assignment $S = \{\overline{B}_0, B_1\}$, GMs with IDs 010 and 011 can decrypt the ciphertext.

1) Encryption: $\mathbf{ENC}(GP, S, M)$ encryption algorithm takes inputs of the group parameter GP, a set of bit-assignment S, the message M, and returns the ciphertext CT. Given the set of bit-assignment S, it is easy to calculate the following terms:

$$\begin{aligned} e(g,g)^{\alpha Y_S} &= e(g,g)^{\alpha \sum_{B \in S} y_B} \\ &= \prod_{B \in S} e(g,g)^{\alpha y_B} \end{aligned}$$

$$\begin{aligned} g^{\beta Y_S} &= g^{\beta \sum_{B \in S} y_B} \\ &= \prod_{B \in S} g^{\beta y_B} \end{aligned}$$

For example, if $S = \{\overline{B}_0, B_1, B_2\}$, $e(g,g)^{\alpha Y_S} = e(g,g)^{\alpha(y_{\overline{B}_0} + y_{B_1} + y_{B_2})}$.

After calculating $e(g,g)^{\alpha Y_S}$ and $g^{\beta Y_S}$, the **ENC** algorithm 2 generates a non-trivial random number $t \in \mathbb{Z}_p^*$. Then, the algorithm computes $C_0 = Me(g,g)^{\alpha t Y_S}$, $C_1 = g^{\beta t Y_S}$, $C_2 = g^t$. Thus, the ciphertext is as:

$$CT: \{S, C_0 = Me(g,g)^{\alpha t Y_S}, C_1 = g^{\beta t Y_S}, C_2 = g^t\}$$

Algorithm 2 **Enc**(MK, S, M)

```
Compute $e(g,g)^{\alpha Y_S} = \prod_{B \in S} e(g,g)^{\alpha y_B}$;
Compute $g^{\beta Y_S} = \prod_{B \in S} g^{\beta y_B}$;
Randomly select $t \in \mathbb{Z}_p$;
Compute $C_0 = Me(g,g)^{\alpha t Y_S}$;
Compute $C_1 = g^{\beta t Y_S}$;
Compute $C_2 = g^t$;
return
$CT: \{S, C_0 = Me(g,g)^{\alpha t Y_S}, C_1 = g^{\beta t Y_S}, C_2 = g^t\}$;
```

2) Decryption: On receiving the CT, GMs whose ID satisfied the bit-assignment S associated with the ciphertext, can decrypt the CT by performing decryption algorithm **DEC**(GP, SK, CT).

The **DEC** algorithm 3 first checks whether the GM u is eligible to decrypt the message by testing whether $S_u \subseteq CT.S$, where $CT.S$ represents the bit assignments associated with the ciphertext CT. Then, for each bit assignment $B \in CT.S$, the algorithm use u's pre-distributed secret shares $D_B = g^{r y_B}$ to compute:

$$\begin{aligned} F &= \prod_{B \in CT.S} g^{r y_B} \\ &= g^{r \sum_{B \in CT.S} y_B} \\ &= g^{r Y_{CT.S}} \end{aligned}$$

Next, the algorithm computes:

$$\begin{aligned} A_1 &= e(C_1, D) \\ &= e(g,g)^{(\alpha + r) t Y_{CT.S}} \end{aligned}$$

and

$$\begin{aligned} A_2 &= e(C_2, F) \\ &= e(g,g)^{r t Y_{CT.S}} \end{aligned}$$

Then the algorithm divides A_1 by A_2 and get

$$\begin{aligned} A_3 &= A_1 / A_2 \\ &= e(g,g)^{\alpha t Y_{CT.S}} \end{aligned}$$

which blinds the plaintext in ciphertext. Finally, the algorithm unblinds the ciphertext by calculating $C_0/A_3 = M$.

E. Encryption for Subgroups of GMs

In this subsection, we present how GC or GMs can securely communicate with arbitrary subgroup of members in an optimal manner. We first define some of the terms used in the following presentations:

- *Literal*: A variable or its complement, e.g., B_1, $\overline{B_1}$, etc.
- *Product Term*: Literals connected by AND, e.g., $\overline{B_2} B_1 \overline{B_0}$.
- *Sum-of-Product Expression (SOPE)*: Product terms connected by OR, e.g., $\overline{B_2} B_1 B_0 + B_2$.

Algorithm 3 **DEC**(GP, SK, CT)

```
if $S_u ! \subseteq CT.S$ then
    return $\perp$;
end if
Compute $F = \prod_{B \in CT.S} g^{r y_B} = g^{r Y_{CT.S}}$;
Compute $A_1 = e(C_1, D) = e(g,g)^{(\alpha + r) t Y_{CT.S}}$
Compute $A_2 = e(C_2, F) = e(g,g)^{r t Y_{CT.S}}$
Compute $A_1/A_2 = A_3 = e(g,g)^{\alpha t Y_{CT.S}}$
Compute $C_0/A_3 = M$
return $M$;
```

Given the subgroup of GMs L, the boolean membership functions $M(B_0, B_1, \ldots, B_{n-2}, B_{n-1})$, which is in the form of SOPE, can determine the membership of the subgroup. That is, only if GM u belongs to the subgroup, $M(ID_u) = 1$. Formally, the following properties of membership functions hold:

$$M(b_0^u, b_1^u, \ldots, b_{n-2}^u, b_{n-1}^u) = \begin{cases} 0 & \text{iff } u \in G \setminus L, \\ 1 & \text{iff } u \in L. \end{cases}$$

For example, if the subgroup $L = \{000, 001, 011, 111\}$, then $M = \overline{B}_0 \overline{B}_1 \overline{B}_2 + \overline{B}_0 \overline{B}_1 B_2 + \overline{B}_0 B_1 B_2 + B_0 B_1 B_2$.

The GC or a GM runs the Quine-McCluskey algorithm [37] to reduce M to minimal SOPE M_{min}. The reduction can consider *do not care* values on those IDs that are not currently assigned to any GM to further reduce the size of M_{min}. Since M_{min} is in the form of SOPE, encryption is performed on each product term. That is, for each product term E in M_{min}, **ENC** algorithm encrypt the message with the set of bit-assignment S that contains all literals in E. The total number of encrypted message equals to the number of product terms in M_{min}.

For example, if $L = \{000, 001, 011, 111\}$, $M_{min} = \overline{B}_0 \overline{B}_1 + B_1 B_2$. We can find that M_{min} contains 2 product terms. the message M for L can be encrypted as $M_{\{\overline{B}_0, \overline{B}_1\}}$ and $M_{\{B_1, B_2\}}$ respectively.

F. GM Leaving

1) Key Update: When several GMs (denoted by set L) are removed from the multicasting group, GC needs to update the $\{MK, GK\}$ as well as the private key for each remaining GM $u \in G \backslash L$. We present how this process can be done efficiently.

GC first changes $MK' = \{\beta, g^{\alpha'}, e(g,g)^{\alpha'}\}$, where α' is randomly selected in $\mathbb{Z}_p$. Then, GC multicasts an encrypted key-update factor $kuf = g^{\frac{\alpha' - \alpha}{\beta}}$. Note that kuf is encrypted, and it cannot be decrypted by any $u \in L$.

Each GM $u \in G \setminus L$ updates its private key SK_u based on the key updating factor $g^{\frac{\alpha' - \alpha}{\beta}}$. This process only updates the component D in SK_u. The new D can be updated by the following method: $D \cdot g^{\frac{\alpha' - \alpha}{\beta}} = g^{\frac{\alpha + r}{\beta}} \cdot g^{\frac{\alpha' - \alpha}{\beta}} = g^{\frac{\alpha + r + \alpha' - \alpha}{\beta}} = g^{\frac{\alpha' + r}{\beta}}$. Also, each $u \in G \setminus L$ updates their GK simply by computing $GK' = H(g^{\frac{\alpha' - \alpha}{\beta}})$.

2) Single or Multiple Leave: We first consider that only one GM leaves the group. For example, if the leaving GM u's ID is 101 with bit-assignment $S_u = \{B_0, \overline{B}_1, B_2\}$. The key updating message is encrypted as $\{kuf\}_{\{\overline{B}_0\}}$, $\{kuf\}_{\{B_1\}}$, $\{kuf\}_{\{\overline{B}_2\}}$ and is multicasted to the entire group. If ID 100 is not assigned, $\{kuf\}_{\{\overline{B}_2\}}$ is not needed. Although the leaving member may intercept the transmitted messages, it cannot decrypt them since every message is encrypted with a bit assignment that the leaving member does not possess. Each of remaining GMs can decrypt at least one of the multicasted messages.

We now focus on the case when multiple GMs leave the multicast group. Given the set of leaving GMs L, GC can easily derive the set of remaining GMs $G \setminus L$ as well as the set of unassigned IDs if GC stores all assigned IDs. If GC does not store assigned ID, GC can assume all IDs are assigned. Then, the GC runs the Quine-McCluskey algorithm [37] to reduce the membership function $M()$ to minimal SOPE. Then, GC can encrypt the key updating factor for each product term. The total number of encrypted key updating factors equals to the number of product terms in M_{min}. For example, we assume that two GMs $\{000, 010\}$ leave, five GMS $\{001, 011, 100, 101, 110\}$ remain, and $\{111\}$ is not assigned to any GM (i.e., the ID bit assignments are *do not care*). With the considerations *do not care* values, M can be reduced to $M_{min} = B_0 + B_2$. GC need to multicast two messages $\{kuf\}_{\{B_0\}}$ and $\{kuf\}_{\{B_2\}}$.

V. Performance Assessments

In this section, we analyze the performance of EGK scheme and compare it with several previous solutions: Flat Table scheme (FT) [1], FT implemented using CP-ABE (FT-ABE) [14], subset-difference broadcast encryption scheme (Subset-Diff) [38], BGW broadcasting encryption [39], access control polynomial (ACP) scheme [40], and Non-Flat-Table tree-based schemes (e.g., OFT [11], LKH [7], ELK [13], etc.). The performance is assessed in terms of communication overhead (number and size of messages incurred by *join* and *leave* operations), storage overhead (group data stored on the GC and GM), and computation overhead (number of cryptographic operations needed in encryption and decryption operations). We denote the group size be N, the number of leaving GMs to be l. Also, for the Subset-Diff scheme, t denotes the maximum number of colluding users to compromise the ciphertext. The summary of comparative results is presented in Table I.

A. Communication Overhead

As a comprehensive comparison, we first discuss the communication overhead of several broadcasting encryption schemes. In Subset-Diff scheme, the communication overhead grows linearly with the maximum number of colluding users to compromise the ciphertext. For BGW scheme, the message size is $O(N^{\frac{1}{2}})$ as reported in [39]. In ACP scheme, the size of message depends on the degree of access control polynomial, which equals to the number of current GMs plus the number of joining GMs or the number of current GMs minus the number leaving GMs. Thus, the message size is $O(N)$.

When removing multiple GMs from EGK group, the number of messages depends on the number of product terms in the M_{min}. In [41], the authors derived an upper bound and lower bound on the average number of products in a minimized SOPE. For example, $\{000, 010\}$ are leaving GMs, and $\{001, 011, 100, 101, 110\}$ are remaining GMs, and $\{111\}$ is not assigned (i.e., *do not care*). In this example, EGK requires 2 messages while tree-based schemes needs at least 3 messages. Now, we prove that EGK achieves storage-communication-optimal:

Lemma 1 (Optimality of EGK): EGK achieves storage-communication-optimal.

Proof Sketch 1: In EGK, each GM is distributed a private key with $\log N$ secret shares and a factor $D = g^{\frac{\alpha+r}{\beta}}$ with constant size. Thus, the storage overhead of EGK is $O(\log N)$. With the $\log N$ secret shares, a GM can combine them to calculate $N-1$ distinct F factors in the DEC Algorithm 3. Thus, each GM can decrypt $N-1$ encrypted streams and EGK is storage-communication-optimal. □

For tree-based multicast key distribution schemes such as OFT [11], LKH [7], ELK [13], etc., the communication overhead for a GM leaving depends on the number of keys in the tree that need to be updated [42], [13]. Some tree-based schemes tried to optimize the number of messages to update all the affected keys in the case of multiple *leaves*. In ELK [13], which is known to be one of the most efficient tree-based schemes, the communication overhead for multiple *leaves* is $O(a-l)$, where a is the number of affected keys and l is the number of leaving GMs. Since there are $\log N$ nodes on the path from root to leaf in the tree structure, the total number of affected keys when l GMs leave the group is $O(l \cdot \log N)$.

In the worst cases, EGK out-performs all the tree-based schemes except Flat-Table. Since EGK requires same number of messages as Flat-Table when removing a set of GMs, we utilize some of the performance results from [1].

Lemma 2 (Worst case of removing 2 GMs [1]): When removing 2 GMs from a group with $N = 2^n$ GMs, the number of key updating messages is at most n. The worst case is achieved when the Hamming distance between 2 GMs is n.

Proof Sketch 2: Please refer to [1]. □

As a comparison, in the same scenario, the number of keys to be updated is $2n-1$, thus ELK requires $2n-3$ messages while EGK requires n messages.

Lemma 3 (worst case of removing multiple GMs [1]): The worst case of removal multiple GMs happens when both of following conditions hold: 1) there are $N/2$ GMs to be removed; 2) the Hamming distance between IDs of any two remaining GMs is at least 2. In the worst case, the number of key updating messages is $N/2$.

Proof Sketch 3: Please refer to [1]. □

In this case, the number of keys to be updated is $N - N/2 = N/2$ for ELK, since there are N non-leaf keys to be updated and the number of leaving GMs is $N/2$. We can see that, in this particular worst case, EGK's performance is same as ELK approach.

In this paper, we simulated EGK along with LKH in groups with 1024 GMs and 4096 GMs, and the number of messages required are shown in Figure 4 and Figure 5 respectively. In the simulation, we consider the cases of 5%, 25%, 50% IDs are not assigned (i.e., *do not care* value). For each case, different percentages of leaving GMs are randomly selected from the

TABLE I

COMPARISON OF COMMUNICATION OVERHEAD AND STORAGE OVERHEAD IN DIFFERENT GROUP KEY MANAGEMENT SCHEMES.

Scheme	Communication Overhead			Storage Overhead	
	join	single *leave*	multiple *leaves*	GC	GM
EGK	$O(1)$	$O(\log N)$	$\approx O(\log N)$	$O(\log N)/O(N)$	$O(\log N)$
Flat-Table	$O(\log N)$	$O(\log N)$	$\approx O(\log N)$	$O(\log N)/O(N)$	$O(\log N)$
Flat-Table-ABE	$O(1)$	$O(\log N)$	$\approx O(\log^2 N)$	$O(\log N)/O(N)$	$O(\log N)$
Subset-Diff	N/A	$O(t \cdot log^2(t) \cdot \log m)$	$O(t \cdot log^2(t) \cdot \log N)$	$O(N)$	$O(log^2(N))$
BGW	N/A	$O(N^{\frac{1}{2}})$	$O(N^{\frac{1}{2}})$	$O(N^{\frac{1}{2}})$	$O(N^{\frac{1}{2}})$
ACP	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(1)$
Non-Flat-Table-Tree	$O(1)$	$O(\log N)$	$O(l \cdot \log N)$	$O(N)$	$O(\log N)$

N: the number of group members; l: the number of leaving members; t: maximum number of colluding users to compromise the ciphertext.

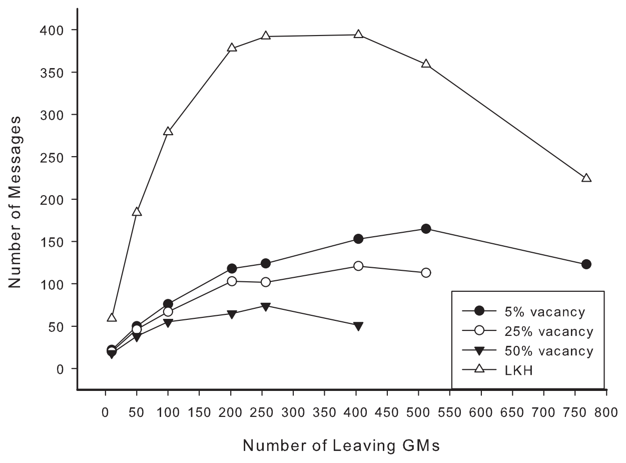

Fig. 4. Number of messages of multiple leave for a group with 1024 GMs.

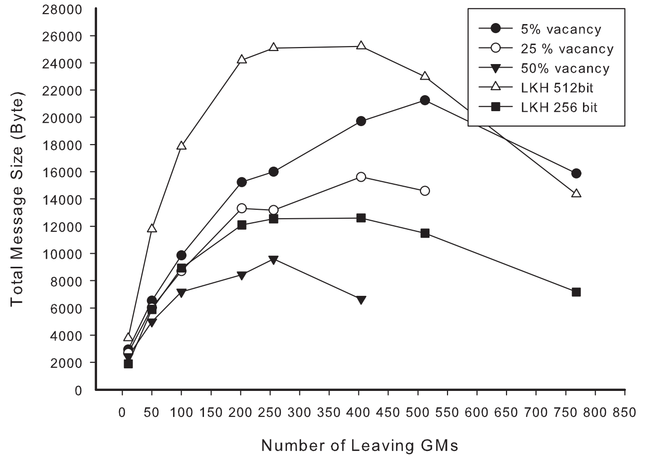

Fig. 6. Size of total messages of multiple leave for a group with 1024 GMs.

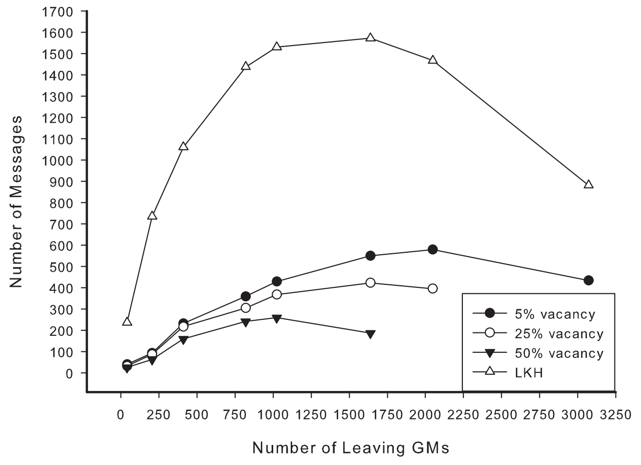

Fig. 5. Number of messages of multiple leave for a group with 4096 GMs.

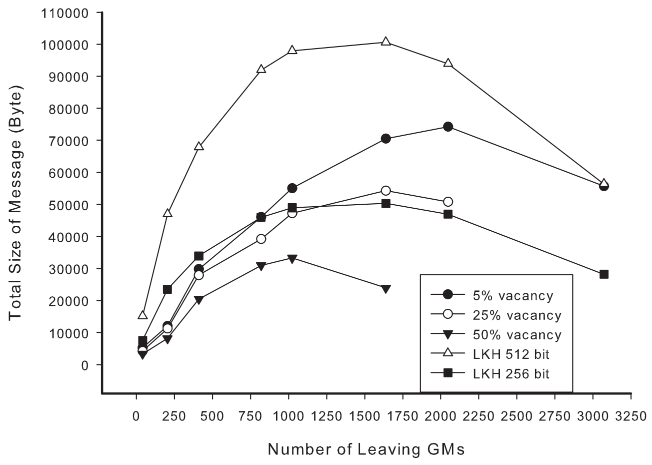

Fig. 7. Size of total messages of multiple leave for a group with 4096 GMs.

group. We repeat 100 times to average the results. As a comparison, the message number curve of LKH is also plotted. From the result, we can see that EGK performs better than LKH and is achieves roughly $O(\log N)$ complexity, where the constant factor is about 20 for the 1024-member group and 50 for the 4096-member group.

Finally, we look into the message size of EGK, FT-CP-ABE[14], and symmetric key tree-based schemes. As mentioned in [14], in FT-CP-ABE, the size of ciphertext grows linearly based on the increase of the number of attributes in the access policy [14], [15]. Experimentally, the message size in FT-CP-ABE starts at about 650 bytes, and each additional attribute adds about 300 bytes. In a system with 10-bit ID or 1024 GMs, the number of attributes using FT-CP-ABE ciphertext is at most 10 and the message size may be as large as 650+9*300=3350 bytes. Since the number of attributes in the access policy is bounded by $\log N$, we can conclude that the communication overhead of FT-CP-ABE is in the order of $O(\log^2 N)$. In EGK, every ciphertext contains exactly one group member on $\mathbb{G}_0$ and two group members on $\mathbb{G}_1$. Empirically, the value on $\mathbb{G}_0$ or $\mathbb{G}_1$ can be about 128 bytes. Thus, the ciphertext in EGK is bounded by 400 bytes, which is significantly smaller than the ciphertext size reported in FT-CP-ABE [14]. Moreover, since the component C_2 in the ciphertext can be shared by multiple messages, we can further reduce the message size of EGK. Existing tree-based schemes using symmetric encryption algorithms, such as AES, enjoys the advantage of small ciphertext. To encrypt a 32-byte GK, those schemes require as low as 32-byte ciphertext. However, based on our evaluation results in Figure 6 and 7, the total message size of EGK will be smaller than symmetric key based schemes when the size of a group is large, thanks to much fewer numbers of transmitted messages. It can be expected in large scale systems, where the size of a multicast group is larger than 4096, EGK will be much more efficient than other schemes.

B. Storage Overhead

In EGK, the storage overhead for GC is $O(\log N)$ if GC does not store IDs of GMs. In this case, GC or GMs can not utilize *do not care* values to further reduce the membership function in SOPE form. Thus, the communication overhead might be higher. The storage overhead is $O(\log N)$ for a GM, since GM stores a private key component for each bit in its ID.

C. Computation Overhead

In this section, we compare the computation overhead of those asymmetric key based schemes. In ACP scheme, the author reports that the encryption needs $O(N^2)$ finite field operations when the sub-group size if N; in the BGW scheme, the encryption and decryption require $O(N)$ operations on the bilinear group, which are heavier than finite field operations [43], [44]. In EGK, each encryption requires $\log N$ operations on the bilinear groups, and the decryption requires 2 pairings. Thus, the complexities of encryption and decryption are bounded by $O(\log N)$ and $O(1)$ respectively. Although the problem of minimizing SOPE is NP-hard, efficient approximations are widely known. Thus, EGK is much more efficient than ACP and BGW when group size is large.

VI. Security Analysis of EGK

In this section, we analyze the security of EGK. Firstly, we note that $Y_S = \sum_{B\in S} y_B$ should be different from $Y_{S'} = \sum_{B\in S'} y_B$, where $S \neq S'$. If there exist S and S' ($S \neq S'$) such that $Y_S = Y_{S'}$, a GM with bit-assignments S' will be able to decrypt the ciphertext encrypted with bit-assignments S. Remark that the assumption holds with overwhelming possibility $\frac{p(p-1)\cdots(p-N-1)}{P^N} > \frac{(p-N-1)^N}{p^N} = (1-\frac{N-1}{p}) > 1-\frac{N(N-1)}{p} > 1-\frac{N^2}{p}$, where $N = 2^n$ and n is the number of bits in the ID.

Lemma 4: The $\mathbf{ENC}$ and $\mathbf{DEC}$ algorithms are secure given the hardness Discrete Logarithm Problem on $\mathbb{G}_0$ and $\mathbb{G}_1$.

Intuitively, we prove the security of $\mathbf{ENC}$ and $\mathbf{DEC}$ algorithms by showing that there is no attacker who can reveal the ciphertext with non-negligible possibilities, given the intractability of the DLP.

Firstly, we prove that for a non-GM attacker without any predistributed secrets, the possibility that the attacker can recover the ciphertext is $1/p$, where p is the order of large cyclic group $\mathbb{G}_0$ and $\mathbb{G}_1$. Given the public group parameter, it is trivial to derive g and $e(g,g)$. With this information, the possibility that the attacker correctly guesses $\alpha t Y_S$ and recovers M from $M \cdot e(g,g)^{\alpha t Y_S}$ equals to $1/p$, since the order of $\mathbb{G}_1$ is p and DLP is hard on $\mathbb{G}_1$.

Secondly, given a GM u with a set of bit-assignments S_u and a ciphertext CT, the possibility that u can decrypt the ciphertext equals to $1/p$, if the $CT.S \nsubseteq S_u$. The possibility that the attacker can guess correct $g^{rY_{CT.S}}$ with is at most $\frac{N^2}{p}$, when $Y_{CT.S} = Y_{S'}$ for some $S' \neq CT.S$.

Thirdly, if an attacker possesses private keys generated by a system master key MK, the possibility that the attacker can decrypt the message encrypted under different system master key MK' is $1/p$. Suppose the private key of attacker is generated with α_1 and one M is encrypted by α_2, we have $Me(g,g)^{\alpha_2 t Y_S}$. Now, if the GM tries to decrypt the message, the result he/she derived is $M \cdot e(g,g)^{(\alpha_2-\alpha_1)tY_S}$. Given the randomness of α, the possibility that $\alpha_1 = \alpha_2$ equals to $1/p$.

In summary, given the hardness of DLP on $\mathbb{G}_0$ and $\mathbb{G}_1$, we can prove the possibility that an attacker can break the $\mathbf{ENC/DEC}$ algorithms with $1/p$ or $\frac{N^2}{p}$ possibilities, where p is a very large integer and it produces a negligible probability $1/p$. □

Lemma 5: EGK provides backward secrecy.
When new GMs *join* the group, a new random GK' is encrypted ($\{GK'\}_{GK}$) and multicasted. Also, the private key of joining GMs are generated under a random system master key α_1. We note that all the previous key updating factors are encrypted using different random α's. Suppose one of the key-update factors M are encrypted by α_2, we have $Me(g,g)^{\alpha_2 t Y_S}$. Now, if the GM tries to decrypt the message, the result he/she derived is $Me(g,g)^{(\alpha_2-\alpha_1)tY_S}$.

Given 1) randomness of GK'; 2) the randomness of $\alpha's$; 3) security of symmetric encryption and 4) Discrete Logarithm (DL) [45] problem on $\mathbb{G}_0$ and $\mathbb{G}_1$, the new joining group member cannot derive any previous GKs or key-update factors. Thus, the group backward secrecy is guaranteed. □

Lemma 6: EGK provides group forward secrecy.
When GMs *leave* the group, the GC updates the system parameters to MK' using a new random α' and multicasted the encrypted $g^{\frac{\alpha'-\alpha}{\beta}}$ to all the GMs in $G \setminus L$. All remaining GMs will update their private keys and GK using the key updating factor $g^{\frac{\alpha'-\alpha}{\beta}}$. Based on Lemma 1, the leaving GMs can not decrypt the key update factor. Thus, the leaving GMs cannot decrypt future encrypted messages since GK has been changed.

Even if a leaving GM stores all encrypted key-updating messages and *join* the group again, he or she cannot decrypt a previous key updating message, since these messages are encrypted under different master keys, which is intractable to the joining GM based on Lemma 1. Moreover, using the key updating factor $g^{\frac{\alpha'-\alpha}{\beta}}$ to derive g^{α} and β is hard due to the DL problem. □

Lemma 7: Leaving GMs cannot collude to decrypt multicasted messages targeted to other GMs.
We refer to the collusion attack as any combinations of GMs attempting to derive other GM's private key by combining their private keys. We first show that any two GMs cannot collude using their private keys. Given the private keys of two attackers a_1 and a_2, $SK_{a_1} = \{D = g^{\frac{\alpha+r_{a_1}}{\beta}}, \forall B \in S_{a_1} : D_B = g^{r_{a_1} y_B}\}$, $SK_{a_2} = \{D = g^{\frac{\alpha+r_{a_2}}{\beta}}, \forall B \in S_{a_2} : D_B = g^{r_{a_2} y_B}\}$, the problem of deriving $SK_v = \{D = g^{\frac{\alpha+r_v}{\beta}}, \forall k \in SA_v : D_A = g^{r_v y_k}\}$, where $S_v \subseteq S_{a_1} \bigcup S_{a_2}$ and $S_v \neq S_{a_1}$ $S_v \neq S_{a_2}$, can be reduced to the Discrete Logarithm problem on $\mathbb{G}_0$. Furthermore, adding more colluding attackers will not help due to the hardness of DL problem. □

VII. Discussion

A. Clustering for Scalability

In the previous sections, we introduce the basic construction of EGK, where the size of max group members is limited by the size of ID space. If the ID space is restricted by n bits, the size of the overall group is limited to 2^n. Moreover, since the problem of boolean function minimization is NP-hard and the run time of QuineMcCluskey algorithm grows exponentially with the number of variables, the number of bits in ID cannot be very large. To accommodate potentially much larger member population, we can divide the multicast group into several clusters and each cluster has a cluster controller (CC). The dynamic membership of CC is managed by GC; GC and CCs will setup a shared key K_g, which is only shared among GC and CCs. On the other hand, each cluster is managed by its CC and dynamic membership is handled locally. GMs and CC in the cluster c_i will share a cluster key K_{c_i}. The multicast group including GMs, GC, and CCs share a unique GK to protect multicast data. In Fig. 8, we depict the clustering of EGK and the allocation of shared keys.

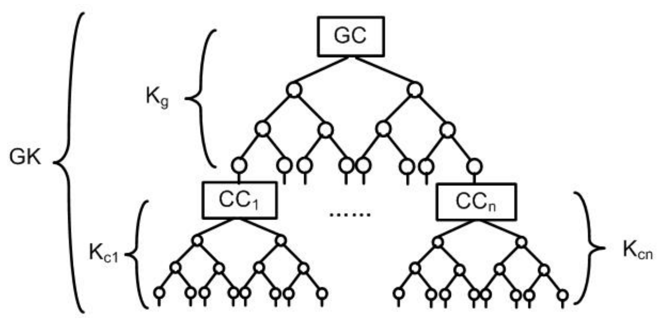

Fig. 8. Illustration of clustering in EGK to accommodate a large member population.

Add/Delete GMs: When a GM joins the group, it will be assigned to a cluster c_i and its CC will perform the group member addition operations described in IV-C to issue the GM a unique ID within this cluster and private keys. GK and K_{c_i} are updated correspondingly as in IV-C. A new GK' is generated and encrypted by current GK while a new K'_{c_i} is generated and encrypted by current K_{c_i}.

Note that each cluster's master key can be independent. In this way, the GMs' private keys are local effective. Thus, in the event of removing GMs from a cluster, key update operation is only performed locally. The deletion of GMs is identical to the section IV-F, except that the new K_{c_i} needs to be updated by each cluster c_i.
Add/Delete Clusters: Dynamically adding and deleting a cluster is performed identical to the addition and deletion of GMs. When adding a cluster, a cluster controller (CC) is selected and assigned a unique cluster ID, a cluster private key and the K_g shared with the GC. When deleting a cluster, GC communicate will all remaining CCs to update their private keys and the GK for all group members. Note that the deletion of CC means all the GMs in the cluster is excluded from the multicast group.

B. Further Improvement of Efficiency

If we further investigate into the ciphertext, we can reduce the total multicast data size by combining common C_2 components in different encrypted messages targeted for different product terms in the same membership function. For example, if $L = \{000, 001, 011, 111\}$, $M_{min} = \overline{B}_0\overline{B}_1 + B_1B_2$. We can find that M_{min} contains 2 product terms. the message M for L can be encrypted as $M_{\{\overline{B}_0, \overline{B}_1\}}$ and $M_{\{B_1, B_2\}}$. As presented in Section IV-D, the 2 encrypted messages are constructed as

$$\{S_1 = \{\overline{B}_0, \overline{B}_1\}, C_0 = Me(g,g)^{\alpha t Y_{S_1}}, C_1 = g^{\beta t Y_{S_1}}, C_2 = g^t\}$$

and

$$\{S_2 = \{B_1, B_2\}, C_0 = Me(g,g)^{\alpha t Y_{S_2}}, C_1 = g^{\beta t Y_{S_2}}, C_2 = g^t\}$$

Note that the C_2 component in the these 2 messages are identical for the same random t. By combining the C_2, up to 33% of total multicast data can be reduced safely.

VIII. Conclusion

In this paper, we propose a key distribution scheme – EGK that attains the same optimality as Flat Table without collusion vulnerability. EGK also support dynamic subgroup communication initialized by each group members (imagine a virtual chat room in the multicast group). Additionally, EGK provides constant message size and requires $O(\log N)$ storage overhead at the group controller, which makes EGK suitable for applications containing a large number of multicasting group members. Moreover, adding members in EGK requires just one multicasting message. EGK is the first work with such features.

According to the theoretical and experimental analysis, EGK out-performs all existing solutions in the musticast and broadcast application domain. We also discussed expanding scalability of EGK by clustering the multiscast group and further reducing the communication overhead by combining common components in the ciphertext.

References

[1] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, I. Center, and Y. Heights, "Key management for secure lnternet multicast using Boolean functionminimization techniques," *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 1999.

[2] R. Poovendran and J. Baras, "An information-theoretic approach for design and analysis ofrooted-tree-based multicast key management schemes," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2824–2834, 2001.

[3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, I. Center, and Y. Heights, "Multicast security: a taxonomy and some efficient constructions," *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 1999.

[4] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," *Lecture Notes in Computer Science*, pp. 437–452, 2001.

[5] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and secure source authentication for multicast," in *Network and Distributed System Security Symposium, NDSS*, vol. 1. Citeseer, 2001, pp. 35–46.

[6] D. Wallner, E. Harder, and R. Agee, "Key Management for Multicast: Issues and Architectures RFC 2627," *IETF, June*, 1999.

[7] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *Networking, IEEE/ACM Transactions on*, vol. 8, no. 1, pp. 16–30, 2000.

[8] L. Dondeti, S. Mukherjee, and A. Samal, "Scalable secure one-to-many group communication using dual encryption," *Computer Communications*, vol. 23, no. 17, pp. 1681–1701, 2000.

[9] H. Harney, C. Muckenhirn, and T. Rivers, "Group key management protocol (GKMP) architecture," RFC 2094, July 1997, Tech. Rep., 1997.

[10] ——, "Group key management protocol (GKMP) specification," Tech. Rep., 1997.

[11] A. Sherman and D. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, pp. 444–458, 2003.

[12] R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption, Advances in Cryptology-Eurocrypt99," *Lecture Notes in Computer Science*, vol. 1592, pp. 459–474, 1999.

[13] A. Perrig, D. Song, and J. Tygar, "ELK, A New Protocol for Efficient Large-Group Key Distribution," *IEEE SYMPOSIUM ON SECURITY AND PRIVACY*, pp. 247–262, 2001.

[14] L. Cheung, J. Cooley, R. Khazan, and C. Newport, "Collusion-Resistant Group Key Management Using Attribute-Based Encryption," Cryptology ePrint Archive Report 2007/161, 2007. http://eprint. iacr. org, Tech. Rep.

[15] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*, 2007.

[16] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," *Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE98)*, 1998.

[17] X. Li, Y. Yang, M. Gouda, and S. Lam, "Batch rekeying for secure group communications," *Proceedings of the 10th international conference on World Wide Web*, pp. 525–534, 2001.

[18] D. Liu, P. Ning, and K. Sun, "Efficient self-healing group key distribution with revocation capability," *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 231–240, 2003.

[19] W. H. D. Ng, M. Howarth, Z. Sun, and H. Cruickshank, "Dynamic balanced key tree management for secure multicast communications," *IEEE Transactions on Computers*, vol. 56, no. 5, pp. 590–605, 2007.

[20] A. Perrig and J. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks*. Springer, 2003.

[21] J. Fan, P. Judge, and M. Ammar, "Hysor: Group key management with collusion-scalability tradeoffs using a hybrid structuring of receivers," in *Proceedings of the IEEE International Conference on Computer Communications Networks*, 2002.

[22] D. McGrew and A. Sherman, "Key establishment in large dynamic groups using one-way function trees," *Manuscript submitted to IEEE Transactions on Software Engineering. A full version of the paper appears in http://download. nai. com/products/media/nai/misc/oft052098. ps*, 1998.

[23] S. Mittra, "Iolus: A framework for scalable secure multicasting," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4, pp. 277–288, 1997.

[24] R. Safavi-Naini and H. Wang, "New constructions for multicast re-keying schemes using perfect hash families," in *Proceedings of the 7th ACM conference on Computer and communications security*. ACM New York, NY, USA, 2000, pp. 228–234.

[25] Y. Yang, X. Li, X. Zhang, and S. Lam, "Reliable group rekeying: a performance analysis," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM New York, NY, USA, 2001, pp. 27–38.

[26] M. Moyer, J. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *Network, IEEE*, vol. 13, no. 6, pp. 12–23, 1999.

[27] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 309–329, 2003.

[28] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," in *Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE98)*, 1998.

[29] D. Boneh and B. Waters, "A fully collusion resistant broadcast, trace, and revoke system," *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 211–220, 2006.

[30] E. Gafni, J. Staddon, and Y. Yin, "Efficient methods for integrating traceability and broadcast encryption," *Lecture Notes in Computer Science*, pp. 372–387, 1999.

[31] D. Boneh and B. Waters, "A fully collusion resistant broadcast, trace, and revoke system," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, p. 220.

[32] J. Garay, J. Staddon, and A. Wool, "Long-lived broadcast encryption," *Lecture Notes in Computer Science*, pp. 333–352, 2000.

[33] M. Goodrich, J. Sun, and R. Tamassia, "Efficient tree-based revocation in groups of low-state devices," *Lecture Notes in Computer Science*, pp. 511–527, 2004.

[34] D. Halevy and A. Shamir, "The LSD broadcast encryption scheme," *Lecture Notes in Computer Science*, pp. 47–60, 2002.

[35] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," *Lecture Notes in Computer Science*, pp. 41–62, 2001.

[36] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," in *Financial cryptography: 4th international conference, FC 2000, Anguilla, British West Indies, February 20-24, 2000: proceedings*. Springer Verlag, 2001, p. 1.

[37] E. McCluskey, "Minimization of Boolean functions," *Bell System Technical Journal*, vol. 35, no. 5, pp. 1417–1444, 1956.

[38] A. Fiat and M. Naor, "Broadcast Encryption, Advances in Cryptology-Crypto93," *Lecture Notes in Computer Science*, vol. 773, pp. 480–491, 1994.

[39] D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," pp. 573–592, 2006.

[40] X. Zou, Y. Dai, and E. Bertino, "A Practical and Flexible Key Management Mechanism For Trusted Collaborative Computing," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 538–546, 2008.

[41] T. Sasao, "Bounds on the average number of products in the minimum sum-of-products expressions for multiple-value input two-valued output functions," *Computers, IEEE Transactions on*, vol. 40, no. 5, pp. 645–651, May 1991.

[42] J. Snoeyink, S. Suri, and G. Varghese, "A lower bound for multicast key distribution," *Computer Networks*, vol. 47, no. 3, pp. 429–441, 2005.

[43] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[44] A. Ramachandran, Z. Zhou, and D. Huang, "Computing Cryptographic Algorithms in Portable and Embedded Devices," *Portable Information Devices, 2007. PORTABLE07. IEEE International Conference on*, vol. 25-29, pp. 1–7, 2007.

[45] A. Menezes, *Handbook of Applied Cryptography*. CRC Press, 1997.